DETAILED ACTION

1. This action is responsive to the Applicant's response filed 9/30/09.

As indicated in Applicant's response, claims 1-18, 21-22 have been amended. Claims 1-18, 21-22 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 1-18, 21-22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, claims 1, 7, 12 recite generating 'parsed computer code' for a model file; and, in view of the Disclosure, one of ordinary skill in the art would observe that model blocks configuration and form requirements (e.g. required input/output format) pertinent to that configuration cannot be equivalent with a "computer code" being parsed, as much as a model file cannot be same as a *computer code*. That is, as analyzed from the Specifications, content of a model file is parsed by a verification module in order for the latter to generate some knowledge on expected format (para 0040, pg. 11) or information regarding blocks as these are configured within the model (para 0030, pg. 8; para 0025, pg. 6) including input/outputs requirements (para 0028, pg. 7), all of which needed to verify the model file derived requirements against a

previously generated *code_file 103* (para 0026, pg. 6; Fig. 3b). Clearly, no computer code has been parsed **for** the model file, as collection of topographic/structural information/knowledge (e.g. Sum block 308 Fig. 3a) is derived on parsing the *model_file 101* (which is not computer code per se) by the verification module 102. Again, meta-information or format configuration knowledge derived from a model cannot be equated to a "computer code" being parsed **for** the model file. The inventor is not in possession of a generating step, as claimed, where 'parsed computer code' is yielded as one of ordinary skill in the art would not be able to construe how parsing a model file to gather knowledge/format requirements (e.g. about block connectivity) can be same as parsing a computer code to yield "parsed computer code" when only meta type of configuration or format requirements are evidenced/taught from above. The "parsed computer code" will be treated as parsing a model to yield configuration/format knowledge/meta information (e.g. parametric/variable/type requirements, compatibility constraints) no weight being given to 'parsed computer code for the model file'

Claims 2-6, 8-11, 13-18, 21-22 fail to remedy to the lack of description as set forth above, hence are equally rejected as not complying to the written description; e.g. header information in a 'parsed computer code' (refer to claim 5) treated as mere *syntax form* required to construct a class header template or a constructor/signature format.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-18, 21-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al, USPN: 6,983,446 (hereinafter Charisius) in view of APA (Admitted Prior Art: para 0005-0007 of Specifications, pg. 2) according to one of ordinary skill in the art.

Charisius discloses a method for verifying generated computer code having a plurality of lines (e.g. 2102 Fig. 21; 1302 – Fig. 13) generated by a code generating module from a model file of a system including a plurality of functions generated by a model module, the method comprising:

processing the model file (Fig. 13, 15-16, 21-23), by a verification module, to determine values, inputs, outputs (e.g. *Number of Members* – Table 1, *return a value* – Table 11; attributes – Table 12), function type, and syntax for each of the plurality of functions (e.g. *argument ... missing, import declaration* – col. 10 lines 13-21; metrics 1900, 1906 – Fig. 19A, B; Note: audit directives 2110-2112, 2114, 2116, 2118, 2120 – see Fig. 21 to 23 – reads on processing model file – see Fig. 23 -- to derive metrics or requirements – see Table 1 to Table 18 col. 10-20);

generating parsed computer code for the model file (e.g. Fig. 8A; 806 – Fig. 8B; 812 – Fig. 8C; Fig. 19A – Note: information to user as tips, audit acronym/semantics for selection/guidance and code reference for correction guidance – Fig. 8C, 19C -- reads on parsed computer code – see interpretation in **USC 112 Rejection**) based on the determined values, inputs, outputs (see above; see Table 1 to Table 18 col. 10-20), function type, and syntax (see above; see Table 1 to Table 18 col. 10-20) for the model file;

comparing each line of the generated computer code and the parsed computer code (code 2102 – Fig. 21; step 2002, Fig. 20A) to determine if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax (e.g. Fig. 4;

Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13 – Note: audit module to validate first generated code or non-validated source code - Fig. 3, col. 6, lines 12-22 - against requirements and language specifications or audit tips – see Fig. 8A; 806 – Fig. 8B; 812 – Fig. 8C; Fig. 19A; 2110-2112, 2114, 2116, 2118, 2120, Fig. 21 - that encompass rules definitions supported by audit metrics **reads on** comparing lines of first code and second code and, the second code conveyed as audit/verification requirements including underlying support from metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above); and

transmitting an error message if a line of the generated computer code does not include a correct syntax (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67) based on the comparison.

Charisius does not explicitly disclose transmitting the error message *if one or more lines of first generated code does not include a correct value, correct input, correct output, a correct function*. Using a QA module operating on predefined audit error types as set forth above (Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A –col. 16, lines 9-14 – and/or metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above;), Charisius discloses error message to be shown to developers regarding many respects, such as syntax, format parameters of a function, declaring of class or members and this is indicative that any violation determined by a verification or audit process to input/output declarations, or correct values or functions would be raised by the validation process. Programming language syntax in the nature of input/output requirements, type conformance, value conformance would have been evident or

suggested from the above Charisius' presentation of audit rules or metrics and code conformance guidance; while display of error in a validation endeavor such as Charisius's auditing tool such as a visual indication to user about effects from the verifying was a well-known concept as shown above. Well-known practice as in APA, for example, teaches user manually (Background of Invention, Specifications pg. 1-2, para 0004-0007) reviewing errors based on the generated lines of source code written in well-known languages such as C, Pascal, Cobol, Fortran, ADA for programming in the DO-178B simulation framework, e.g. SIMULINK program with lines of source to implement model blocks in terms of programmed input, output, functions. For one of ordinary skill in the art and based on Charisius's validating tool according to well known practice (in software development) of using Charisius' tool to systematically notify the human user/developer using visual means as in APA on any propriety issue regarding code form (e.g. programming correctness as in Ansi C to represent Simulated model with input/output, functions) or implementation thereof, it would have been obvious for one skill in the art at the time the invention was made to implement the error messages by the source code auditing in Charisius so that error message would be displayed when auditing a line of expected source or required format if the line does not include specifically a correct value, correct input, correct output, a correct function as mentioned above. One of ordinary skill in the art would be motivated to do so because not having such proper value or function, correct input or output thereof a code might not properly compile notably when such type of deficiencies would necessarily fall under the types or style of errors contemplated by a auditing tool based on the above.

As per claim 2, Charisius discloses: comparing each line of the generated computer code and the parsed computer code to determine if the generated computer code is missing a line of code (is missing – col 15 lines 47-48; col 25 lines 55-59); and, by virtue of the rationale in claim 1, transmitting the error message if the generated computer code is missing the line of code.

As per claim 3, Charisius discloses the steps of:

comparing each line of the generated computer code and the parsed computer code to determine if the generated computer code includes an extraneous line of code (e.g. col. 20 Table 18; col. 22 lines 61-67); and by virtue of the rationale in claim 1, transmitting the error message if the the generated computer code includes the extraneous line of code.

As per claim 4, Charisius discloses the steps of:

comparing each line of the generated computer code and the parsed computer code to determine if the generated computer code is in a logical order (ordered properly – col. 33, lines 55-60 ; col. 34 lines 10-44); nd by virtue of the rationale in claim 1, transmitting the error message if the first generated computer code is not in the logical order.

As per claim 5, Charisius discloses steps of:

comparing a first header information section in the generated computer code and second header information section in the parsed computer code to determine if the first header information section matches the second header information section (*Declaration* – cols. 25-26; match a declaration, col. 37, lines 1-37- Note: generated source code or class package declaration with respect to expected declaration in OO class or Use case package – see Fig. 14-15, 22 -- in an *audit* instance reads on comparing header of a class signature declaration – see USC 112

Rejection); and (by virtue of the rationale in claim 1) transmitting the error message if the first header information section does not match the second header information section.

As per claim 6, Charisius discloses comparing a first declared variable section in the generated computer code and a second declared variable section in the generated parsed computer code to determine if the first declared variable section matches the second declared variable section (col. 29 lines 7 to col. 30 lines 67; col. 25 lines 20-52 – Note: fixed source code based on positions of declarations, declared type of member with respect to their signature and constructor with respect to a parent class reads on comparing generated variable section with variable section of required format for code); and (by virtue of the rationale in claim 1) transmitting the error message if the first declared variable section does not match the second declared variable section.

As per claim 7, Charisius discloses a computer-readable storage medium containing a set of instructions for verifying a generated computer code having a plurality of lines (refer to claim 1) from a code generating module, the generated computer code automatically generated from a model file of a system having a plurality of functions and created by a model module, the set of instructions comprising:

code that reads in the model file; code that determines values, inputs, outputs, function type, and syntax for each of the plurality of functions in the model file (refer to claim 1);

code that generates a parsed computer code based on the determined values, inputs, outputs, function type, and syntax (refer to claim 1 and USC 112 Rejection);

code that reads in the generated computer code; code that compares each line in the generated computer code and the parsed computer code to determine if the generated computer

code includes the determined values, inputs, outputs, function type, and syntax in the parsed computer code (refer to claim 1); and

code that transmits an error message if a first line in the generated computer code does not include a determined syntax (refer to claim 1) based on the comparison.

Charisius does not explicitly disclose transmitting the error message *if one or more lines of generated code do not include a correct value, correct input, correct output, a correct function*. However, this limitation has been addressed as obvious in claim 1.

As per claim 8, Charisius discloses code that compares each line in the generated computer code and the parsed computer code (refer to claim 7); and code that transmits the error message if the first line does not include the determined value, the determined input, the determined output, the determined function, the determined syntax, or combinations thereof (refer to rationale in claim 1).

As per claim 9, refer to the rationale of claim 2 (line not determined of claim 9 treated as missing line),

As per claims 10-11, refer to claims 4-5

As per claim 12, Charisius discloses a system for verifying the contents of a generated computer code having a plurality of lines generated (refer to first computer code in claim 1) by a code generating module from a model file including a plurality of functions generated by a model module, comprising a processor operable to:

process the model file to determine values, inputs, outputs, function type, and syntax for each of the plurality of functions(refer to claim 1),

generate a parsed computer code for the model file(refer to claim 1 and USC 112 Rejection) based on the determined values, inputs, outputs, functions type, and syntax for the model file,

compare each line in the generated computer code with the parsed computer code to determine if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax(refer to claim 1), and

transmit an error message if a line in the generated computer code does not include a correct syntax based on the comparison(refer to claim 1); and

a display configured to display the error message(refer to claim 1), the display coupled to the processor.

Charisius does not explicitly disclose transmitting the error message if one or more lines of the generated code do not include a correct value, correct input, correct output, a correct function. However, this limitation has been addressed as obvious in claim 1

As per claim 13, refer to claim 2.

As per claim 14, Charisius discloses (by virtue of the obviousness in claim 12) wherein the error message indicates if a line of the first generated computer code has any additional content (e.g. *True ... False literals should not be used ... amount of meaningless code ... use of type casts not necessary ... 'abstract' considered obsolete ... should not use parentheses* – col. 19 lines 47 to col. 20 line 23; col. 16 line 58 to col. 17, line 37).

As per claim 15, Charisius discloses wherein the processor is operable to compare each line in the parsed computer code to the generated computer code to determine if the plurality of lines are in an expected form, and transmit the error message if one or more of the lines of code

in the plurality of lines do not match the expected form (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67).

As per claims 16-18, refer to claim 3-5, respectively.

As per claim 21, Charisius discloses the steps of: comparing each line in the generated computer code and the parsed computer code to determine if the plurality of lines are complete (col. 16 line 55 to col. 17 line 37; avoid empty catch block … without empty body – Table 17 col. 18-19); and transmitting an error message (refer to obviousness rationale in claim 1) if the plurality of lines are incomplete.

As per claim 22, Charisius discloses wherein the set of instructions further comprises: code that compares each line in the generated computer code to the parsed computer code to determine if the generated computer code is complete(refer to claim 21); and code that transmits the error message if the generated computer code is incomplete (refer to claim 21).

Response to Arguments

6. Applicant's arguments filed 9/30/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC 35 § 103 Rejection

(A) Applicants have submitted that simultaneously updating model and viewing of updated source code by Charisius -- so that model and code are consistent with each other -- does not disclose "parsed computer code" (or two codes) from a same model file as claimed (Appl. Rmrks, pg. 10, top para). The 'parsed computer code' in view of the Disclosure cannot be construed as 'computer code' resulting from parsing computer code; and in terms of more

specific support by the Disclosure (refer to the USC 112 Rejection) the inventor is not deemed having possession of this feature recited as generating from a model file a computer code referred to as 'parsed computer code'. Interpretation of this feature amounts to gathering of relationship constructs or topographic information captured from parsing model file such as relationships among blocks therein. Charisius discloses automatically generate source code relevant to a parsing a particular UML file, and using the graphical tool provide the pertinent meta-information for the code constructs for that model in terms of source code auditing requirements; hence Charisius discloses generating a model-derived source code and meta-requirements or auditing additional knowledge pertinent to syntax as to how to properly implement the source code, i.e. another form of *information* (or "parsed computer code" as per USC 112 Rejection) resulting from the same model. The argument is not persuasive given the lack of support from the Disclosure.

(B) Applicants have submitted that knowledge from one of ordinary skill in the art does not teach or suggest 'generating a parsed computer code ... based on determined values ... model file' and 'comparing each line ... and parsed computer code … to determine ... correct syntax' (Appl. Rmrks pg. 10, bottom half). Admitted prior art or well-known prior practice understood by one of ordinary skill in the art at the time the invention was made constitute prior art or knowledge that would complement the teaching or suggestion identified in the base reference. One of ordinary skill in the art would understand the use of programming language with syntax requirements as in Charisius to implement model, and model like Simulink (e.g. as by APA) effectuated via user interface where lines of source code (in the likes of C, Ada, Fortran languages) were well known to represent the blocks relationship of the target model whereby

correctness of code would have to be analyzed and presented for evaluation to the user. The Applicants have chosen not to point out where well-known practices under the skill level of and integration by OSA, would not render obvious the specific subject matter (i.e. *error message if one or more lines of the generated code do not include a correct value, correct input, correct output, a correct function*) being particularly addressed in the 103 Rationale. The argument is deemed insufficient.

(C) Applicants have submitted that (for claims 7-11, 22)the subject matter of 'parsed computer code based on determined values, inputs, outputs function type and syntax' and 'code that compares ... to determine ... syntax in the parsed computer code' was not met (Appl. Rmrks pg. 11, middle) because Charisius in combination with OSA fail to suggest and teach every element recited for the same reasons set forth in claim 1. The rejection will stand in view of the reply presented in sections A, and B from above, in light of the ramification engendered by the USC 112 Rejection.

(D) Applicants have submitted that for claims 12-18, Charisius and OSA fail to teach or suggest every element of the language recited as 'generate ... parsed computer code based on determined values, inputs, outputs function type and syntax' and 'compare each line ... to determine ... syntax in the parsed computer code' (Appl. Rmrks pg. 12, top). Similarly, the argument is referred to the response set forth in sections A, B from above.

In all, the claims stand rejected as set forth in the Office Action.

(E) Applicants, by way of an *Interview Summary*, have submitted that (Appl. Rmrks pg. 12) the Examiner had indicated that *metadata* was used to compare with first generated code in the course of the verification process. Indeed, the weight of the subject matter that would be deemed

more appropriate in representing the invention revolves around how to present such "metadata" in the claims. Following up on the above, the Examiner hereby advises that the structured language using shorthand, tags, along with labels to represent summation/input/outputs of blocks as disclosed (e.g. para 0032-0036) constitutes one such example of representative language that as claimed subject matter, would more favorably affect the course of prosecution of the case.

Conclusion

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

January 06, 2010